Advanced Systems Lab
Tutorial IV
Queuing Systems

G. Alonso
Systems Group
http://www.systems.ethz.ch
Queuing system
Characterizing a queuing system

- Arrival rate
- Service time
- Service discipline
- System capacity
- Number of servers
- Population size

- $A/S/m/C/P/SD$
  - $A$ = arrival distribution
  - $S$ = service distribution
  - $m$ = number of servers
  - $C$ = buffer capacity
  - $P$ = population size (input)
  - $SD$ = service discipline

- Notation not standardized
ARRIVAL RATE DISTRIBUTION
The interarrival times are assumed to form a sequence of Independent and Identically Distributed (IID) random variables. A common assumption is a Poisson distribution.
Mean arrival rate

• Mean interarrival time: $E[\tau]$
• Mean arrival rate: $\lambda = 1 / E[\tau]$
• $\lambda$ is not a random variable!
• Examples:
  – a single client submits a query every 200 ms, then $\lambda$ is 5 queries/second
  – 10 clients submit a query each every 500 ms, then $\lambda$ is 20 queries/second
  – These are the queries submitted to the system
Assumptions

• Queuing systems assume an arrival rate
  – state independent (does not depend on number of previous jobs)
  – stationary (does not change in time)

• These assumptions do not hold in real systems
  – Burstiness / batch jobs
  – Flash crowds (popularity)
  – Social effects (time of day variant load)
SERVICE RATE DISTRIBUTION
Service time per job

• The time it takes to process a job (only the time it takes to process it, not including the time it has been waiting in the queue) = \( s \)

• Mean service rate: \( \mu = 1/E[s] \)

• If there are \( m \) servers, mean service rate is \( m\mu \)

• \( \mu \) is not a random variable!

• Example:
  – Printer takes on average 20 seconds per job, then \( \mu = 0.05 \) jobs/second = 3 jobs/minute
Throughput

- Sometimes $\mu$ is called the system’s throughput
- Careful with the notion of throughput
- This is correct only in some cases
  - There are always jobs ready when a job is finished
  - No overhead in switching to new job
  - All jobs complete correctly
  - Service rate is state independent (does not depend on the number of jobs in the queue)
  - Service rate is stationary (does not change with time)
SERVICE DISCIPLINE
Queue discipline

- FCFS = First Come – First Served
  - Ordered queue
- LCFS = Last Come – First Served
  - Stack
- RR = Round Robin
  - CPU allocation to processes
- RSS = random
- Priority Based
OTHER PARAMETERS
System capacity

• The system (or buffer) capacity is the maximum number of jobs that can be waiting for service
• System capacity includes jobs waiting and jobs receiving service
• In reality = finite
• Analysis = assume infinite capacity
• Finite buffers very important in practice
Number of Servers

• The service can be provided by one or more servers
• Assume work in parallel and independent
• Servers do not interfere with each other
• Total service rate is aggregation of each individual service rate
Population

• The total number of potential jobs that can be submitted to the system:
• Analysis = assume infinite
• In practice:
  – Very large (assume infinite), e.g., number of clicks on a page
  – Finite, number of homework submissions for this lecture
  – Closed systems (output determines input)
GENERAL RESULTS
G/G/1    G/G/m
Offered load

• The offered load or traffic intensity is
\[ \rho = \frac{\lambda}{(m \mu)} = \lambda \cdot \frac{E[s]}{m} \]

• The system is stable if
\[ \rho < 1 \Rightarrow \lambda < m \mu \]

• In other words, the system is stable if the mean arrival rate (\( \lambda \)) is less than the mean service rate (\( m \mu \)), otherwise the queue grows without bounds
\[ \rho = 1 \]

- Unless arrivals and service are deterministic and exactly scheduled, \( \rho = 1 \) does not lead to a steady system
  - Randomness prevents queuing from emptying
  - Server cannot catch up
  - Queue grows without bound

- One way to avoid this is **flow control** (drop jobs when load to high)
Examples

Instrumentation shows that a disk is serving 50 I/O operations per second and the average I/O time is 10 ms. What is the disk utilization?

\[ \rho = \lambda \cdot \frac{E[s]}{m}, \text{ with } m = 1 \]

\[ \rho = 50 \times 0.010 = 0.5 = 50\% \]

Application A generates about 50 I/O requests/s, if the disk is 85 % utilized, what is the average time needed for every I/O? ... 17ms
Further examples

We have allocated 60% of the disk to one application. If we want to maintain an average response time for every I/O operations of 12 ms, what is the maximum number of I/O requests per second that the application can generate?

\[ 0.6 = \lambda \cdot 12 \text{ ms} \Rightarrow \lambda = 50 \text{ req/s} \]
Some more notation

- \( n = n_s + n_q \), where
  - \( n \) is the number of jobs in the system (queue)
  - \( n_s \) is the number of job in the service
  - \( n_q \) is the number of jobs waiting for service

- \( w = w_q + s \), where
  - \( w \) is the total time in the system
  - \( w_q \) is the time waiting in the queue
  - \( s \) is the time in the service

- These are all random variables
Little’s Law

• For the queuing system:

\[ E[n] = \lambda \cdot E[w] \]

• For the queue

\[ E[n_q] = \lambda \cdot E[w_q] \]

• With

\[ E[n] = E[n_q] + E[n_s] \]
\[ E[w] = E[w_q] + E[s] = E[w_q] + \frac{1}{\mu} \]
Example

Instrumentation shows that the average time to respond to a request was 100 ms and the server received about 100 requests/second. If each active request requires 5 KB of memory, how much memory needs to be reserved for the average number of requests in the system?

Jobs in the system = 100 \times 0.1 = 10 \text{ jobs} \ (\text{Little's})

Memory needed = 10 \times 5 \text{ KB} = 50 \text{ KB}
Jobs in service

• Using Little’s Law, one can derive:
  \[ E[n_s] = \frac{\lambda}{\mu} = \lambda \cdot E[s] \]

• For a queuing system with m servers
  \[ E[n_s] = m \cdot \rho \]

that is, the average number of jobs in service is m times the arrival rate divided by the mean service rate
BIRTH-DEATH PROCESSES
Stochastic processes

• Many of the values in a queuing system are random variables function of time (e.g., the waiting time at a queue)
• Such random functions of times are called stochastic processes
• If the values a process can take are finite or countable, it is a discrete process or a stochastic chain
Markov Processes

• If the future states of a process depend only of the current state and not on past states, the process is a Markov process
• Discrete Markov processes are Markov chains
• A Markov chain in which the transition between states is limited to neighboring states is called a birth-death process
Steady state probability

- Probability of being in state $n$ is:

$$p_n = \frac{\lambda_0 \lambda_1 \ldots \lambda_{n-1}}{\mu_1 \mu_2 \ldots \mu_n} \cdot p_0$$
M/M/1
M/M/1

- Memoryless distribution for arrival and service
- Single server
- Infinite buffers and FCFS
- Mean arrival rate: $\lambda$
- Mean service rate: $\mu$

\[
\begin{array}{c}
\lambda \\
0 \\
\mu \\
\lambda \\
1 \\
\mu \\
\lambda \\
2 \\
\mu \\
\lambda \\
\end{array}
\]

\[
\begin{array}{c}
\lambda \\
i \\
\mu \\
\lambda \\
\mu \\
\lambda \\
\mu \\
\end{array}
\]
Basics M/M/1

• From the state probability of a birth-death process:
  \[ p_n = \left(\frac{\lambda}{\mu}\right)^n p_0, \quad n = 1, 2, \ldots, \infty \]

• or
  \[ p_n = \rho^n p_0 \]

• Since all probabilities must add to 1
  \[ p_0 = 1 - \rho \quad \text{and} \quad p_n = (1 - \rho) \rho^n \]
Utilization

- Utilization: probability that there is one or more jobs in the system

\[ U = 1 - p_0 = \rho \]
M/M/1 behavior

• The mean number of jobs $E[n]$ is

$$E[n] = \sum_{n=1}^{\infty} n \cdot p^n = \sum_{n=1}^{\infty} n(1-\rho)\rho^n = \frac{\rho}{1-\rho}$$

• Applying Little’s Law we get the response time

$$E[w] = \frac{1/\mu}{1-\rho}$$
Response time in M/M/1
M/M/m and M/M/m/m/B
• Each server serves $\mu$ jobs per unit of time
• Jobs get service right away if less than $m$ jobs in system, otherwise they wait in queue.
Probabilities in M/M/m

- Number of jobs in a M/M/m system is a birth death process. Hence:

\[ p_n = \frac{\lambda_0 \lambda_1 \cdots \lambda_{n-1}}{\mu_1 \mu_2 \cdots \mu_n} p_0 \]
Resolving for M/M/M/m

\[ p_n = \frac{\lambda^n}{n! \mu^n} p_0 \]  \hspace{1cm} \text{for } n = 0, 1, 2, \ldots, m-1

\[ p_n = \frac{\lambda^n}{m! m^{n-m} \mu^n} p_0 \]  \hspace{1cm} \text{for } n = m, m+1, \ldots, \infty
With traffic intensity

- $\rho = \frac{\lambda}{m \mu}$

\[ p_n = \frac{(m \rho)^n}{n!} p_0 \]  for $n = 0, 1, 2, \ldots, m-1$

\[ p_n = \frac{\rho^n m^m}{m!} p_0 \]  for $n = m, m+1, \ldots, \infty$
M/M/m queue

• The rest of the results for this system can be derived from the state probabilities (see in the text book)
m times M/M/1 vs M/M/m

• What is better?
  – m queues of the form M/M/1 with arrival rate \( \lambda/m \)
  – a single system of the form M/M/m with arrival rate \( \lambda \)

• In general, M/M/m will be better because it leads to less waiting time (jobs waiting in a queue do not benefit if a server in another queue is free)
M/M/m/B

- The queuing system has limited buffer capacity. After B buffers are full, jobs are no longer admitted.
- State transition diagram is similar to that of M/M/m but it finishes with B (as opposed to having $\infty$ states).
- As before:

$$p_n = \frac{\lambda_0 \lambda_1 \ldots \lambda_{n-1}}{\mu_1 \mu_2 \ldots \mu_n} p_0$$
State probabilities

\[ p_n = \frac{\lambda^n}{n!} \mu^n \quad p_0 = \frac{(m\rho)^n}{n!} \quad p_0 \]

for \( n < m \)

\[ p_n = \frac{\lambda^n}{m!m^{n-m}} \mu^n \quad p_0 = \frac{\rho^n m^m}{m!} \quad p_0 \]

for \( n = m, m+1, \ldots, B \)
M/M/m/B

- All the other parameters can be computed from these probabilities (see the textbook)

- Effective arrival rate:
  - Arrival rate $\lambda$
  - After B jobs, no more jobs enter the system
  - $\lambda' = \lambda (1-p_B)$ effective arrival rate
  - $\lambda - \lambda' = \lambda p_B$ packet loss rate

- Apply effective arrival rate to Little’s Law